

Tablice: Liczby

Informatyka

Jolanta Bachan
2008-04-24

Tablice i listy danych

- Lista to uporządkowany zbiór danych skalarnych. Tablica to zmienna, która przechowuje listę. Każdy *element* tablicy to odrębna zmienna skalarna z niezależną wartością skalarną. Te wartości są uporządkowane, tzn. mają odpowiednią kolejność od najniższego to najwyższego elementu.
- Tablice mogą mieć dowolną ilość elementów. Najmniejsza tablica ma zero elementów.

Reprezentacja literalna

- Literał listowy

- sposób w jaki wartość listy jest reprezentowana w programie;
- jest zbudowana z wartości oddzielonych przecinkiem, zawartych w nawiasach okrągłych. Te wartości tworzą elementy listy.

(1,2,3) tablica z wartościami 1, 2, i 3

("fred",4.5) – dwie wartości, "fred" i 4.5

Reprezentacja literalna

- Elementy listy nie muszą być stałymi.

$(\$a, 17)$ – dwie wartości: obecna wartość $\$a$,
i 17

$(\$a+\$b, \$d+\$e)$ – dwie wartości

Reprezentacja literalna

- Pusta lista (z zero elementami) jest reprezentowana przez pustą parę nawiasów.

`()` – the empty list with zero elements

Funkcja konstruująca listę

(1..5) – same as (1,2,3,4,5)

(2..6,10,12) – same as (2,3,4,5,6,10,12)

(\$a..\$b) – range determined by the current values of \$a and \$b

Zmienne

- Zmienna tablicowa przechowuje pojedynczą listę (zero lub więcej wartości skalarnych)
- Zmienna tablicowa zaczyna się od @.

@fred – is the array variable @fred

@A_Very_Long_Variable_Name

Zmienne

- Zmienna tablicowa przechowuje pojedynczą listę (zero lub więcej wartości skalarnych)
- Zmienna tablicowa zaczyna się od @.

@fred – is the array variable @fred

@A_Very_Long_Variable_Name

- Zmienna tablicowa @fred jest niezależna od zmiennej skalarnej \$fred.

Operator przypisania tablicy

```
@fred = (1,2,3) ;
```

```
@barney = @fred ;
```

Operator przypisania tablicy

```
@fred = (1,2,3) ;
```

```
@barney = @fred ;
```

Jeśli przypiszesz wartość skalarną do zmiennej tablicowej, to wartość skalarna stanie się pojedynczym elementem tablicy:

```
@huh = 1 ; 1 is promoted to the list  
(1) automatically that is, @huh now  
is (1)
```

Operator przypisania tablicy

@fred = (3,4,5) ;

@barney = (1,2,@fred,6) ; @barney
becomes (1,2,3,4,5,6)

@barney = (0,@barney) ; - puts 0 in
front of @barney

@barney = (@barney,7) ; - puts 7 at
the end of @barney

@barney is now (0,1,2,3,4,5,6,7)

Operator przypisania tablicy

$(\$a, \$b, \$c) = (1, 2, 3)$; - give 1 to \$a, 2 to \$b, 3 to \$c

$(\$a, \$b) = (\$b, \$a)$; - swap \$a and \$b

$(\$d, @fred) = (\$a, \$b, \$c)$; - give \$a to \$d and (\$b, \$c) to @fred

$(\$e, @fred) = @fred$; - remove the first element of @fred to \$e, this makes @fred = (\$c) and \$e = \$b

Długość tablicy

- Aby otrzymać wielkość tablicy, należy przypisać zmienną tablicową do zmiennej skalarnej.

```
@fred = (1,2,3) ;
```

```
$a = @fred ; - $a gets 3, the  
current length of @fred
```

Długość tablicy

- Aby otrzymać wielkość tablicy, należy przypisać zmienną tablicową do zmiennej skalarnej.

```
@fred = (1,2,3) ;
```

```
$a = @fred ; - $a gets 3, the  
current length of @fred
```

NAUCZCIE SIĘ TEGO!!!

Dostęp do elementów tablicy

- Elementy tablicy są ponumerowane przy pomocy kolejnych liczb całkowitych, zaczynając od 0, i zwiększając się o 1 z każdym elementem. Dostęp do pierwszego elementu tablicy `@fred` jest uzyskiwany poprzez `$fred[0]`.

Dostęp do elementów tablicy

- Elementy tablicy są ponumerowane przy pomocy kolejnych liczb całkowitych, zaczynając od 0, i zwiększając się o 1 z każdym elementem. Dostęp do pierwszego elementu tablicy `@fred` jest uzyskiwany poprzez `$fred[0]`.



Dostęp do elementów tablicy

- Elementy tablicy są ponumerowane przy pomocy kolejnych liczb całkowitych, zaczynając od 0, i zwiększając się o 1 z każdym elementem. Dostęp do pierwszego elementu tablicy `@fred` jest uzyskiwany poprzez `$fred[0]`.

 Zmienna skalarna
(element tablicy)

 Zmienna tablicowa

Dostęp do elementów tablicy

```
@fred = (7,8,9) ;
```

```
$b = $fred[0] ; - give 7 to $b  
  (first element of @fred)
```

```
$fred[0] = 5 ; now @fred = (5,8,9)
```

Dostęp do elementów tablicy

```
@fred = (5,8,9) ;
```

```
$c = $fred[1] ; - give 8 to $c
```

```
$fred[2]++ ; - increment the 3rd  
element of @fred
```

```
$fred[1] += 4 ; add 4 to the 2nd  
element
```

```
($fred[0], $fred[1]) = ($fred[1],  
$fred[0]) ; swap the first two
```

Dostęp do elementów tablicy

```
@fred = (7,8,9) ;
```

```
@fred[0,1] ; - same as ($fred[0],  
$fred[1])
```

```
@fred[0,1] = @fred[1,0] ; - swap the  
first two elements
```

```
@fred[0,1,2] = @fred[1,1,1] ; - make  
all 3 elements like the 2nd
```

```
@fred[1,2] = (9,10) ; - change the  
last two values to 9 and 10
```

Dostęp do elementów tablicy

```
@fred = (7,8,9) ;
```

```
$a = 2 ;
```

```
$b = $fred[$a]; - like $fred[2] or 9
```

```
$c = $fred[$a-1]; - $c gets  
$fred[1], or 8
```

```
($c) = (7,8,9)[$a-1] ; ($c) gets 8
```

Dostęp do elementów tablicy

```
@fred = (7,8,9) ;
```

```
@barney = (2,1,0) ;
```

```
@backfred = @fred[@barney] ; # same  
as @fred[2,1,0], or ($fred[2],  
$fred[1], $fred[0])
```

Dostęp do elementów tablicy

`$#fred` służy do otrzymania wartości indeksu ostatniego elementu tablicy `@fred`.

```
@fred = (7,8,9) ;
```

```
$last_subscript = $#fred ;
```

```
print "This is the last subscript:  
$last_subscript" ;
```

Dostęp do elementów tablicy

Wartości ujemne indeksu tablicy odliczają elementy od końca. Zatem innym sposobem dostępu do ostatniego elementu jest użycie indeksu -1. Dostęp do przedostatniego elementu byłby za pomocą -2, itd.

```
@fred = (7,8,9) ;  
print $fred[-1] ; # prints 9  
print $#fred ; # prints 2  
print $fred[$#fred] ; # prints 9
```

Funkcje push i pop

- Tablice są często używane jako stosy danych, w których nowe wartości są dodawane lub usuwane z prawej strony listy.

Funkcje push i pop

```
@mylist = (1,2,3) ;  
$new_value = 4 ;  
push(@mylist, $new_value) ;  
  # like @mylist = (@mylist,  
  $new_value)  
$old_value = pop(@mylist) ; #  
  removes the last element of @mylist  
push (@mylist, 4,5,6) ; # @mylist =  
  (1,2,3,4,5,6)
```

Funkcje `shift` i `unshift`

- Funkcje `shift` i `unshift` wykonują operacje na lewej stronie listy (elementach z najniższymi indeksami).

Funkcje `shift` i `unshift`

- Funkcje `shift` i `unshift` wykonują operacje na lewej stronie listy (elementach z najniższymi indeksami).

```
unshift(@fred, $a) ;  
# like @fred = ($a, @fred)
```

```
unshift(@fred, $a, $b, $c) ;  
# like ($a, $b, $c, @fred)
```

```
$x = shift(@fred) ; # like  
($x, @fred) = @fred
```

Funkcje `shift` i `unshift`

- Funkcje `shift` i `unshift` wykonują operacje na lewej stronie listy (elementach z najniższymi indeksami).

```
@fred = (5,6,7) ;
```

```
unshift(@fred,2,3,4) ; # @fred is  
now (2,3,4,5,6,7)
```

```
$x = shift(@fred) ; # $x gets 2,  
@fred is now (3,4,5,6,7)
```

Funkcja reverse

- Funkcja reverse odwraca kolejność elementów listy, zwracając powstałą listę.

```
@a = (7, 8, 9) ;
```

```
@b = reverse(@a) ; # gives @b the  
value of (9, 8, 7)
```

```
@b = reverse(7, 8, 9) ; # same thing
```

Funkcja sort

- Funkcja `sort` bierze argumenty i uporządkowuje je jakoby były pojedynczymi łańcuchami we wzrastającym porządku ASCII. Funkcja ta zwraca uporządkowaną listę, bez zmieniania listy oryginalnej.

```
@x = (1, 2, 3, 6, 14, 25, 30) ;
```

```
@y = sort(@x) ; # @y gets  
(1, 14, 2, 25, 3, 30, 6)
```

<STDIN> jako tablica

- W kontekście listowym, <STDIN> zwraca wszystkie wiersze aż do końca pliku. Każdy wiersz jest zwracany jako osobny element lisy.

@a = <STDIN> ; - read standard input
in a list context

- Wpisz CTRL-Z, aby wskazać koniec pliku.
- Każdy element to łańcuch zakończony nową linią, która jest równoważna z końcową linią wpisanego wiersza.

Ćwiczenia

1. Napisz program, który czyta listę liczb w oddzielnych liniijkach i drukuje listę w odwrotnej kolejności. (Podpowiedź: Prawdopodobnie będziesz musiał wpisać CTRL-Z, aby wskazać koniec listy.)
2. Napisz program, który czyta liczbę, potem listę liczb w oddzielnych liniijkach i drukuje liniijkę z listy, wskazaną przez liczbę podaną na początku.

Do zobaczenia za tydzień!